CLAIMS

What is claimed is:

1. A method, comprising:

receiving active thread state of a first active thread using a virtual state mechanism;

generating virtual thread state in accordance with the active thread state of the first active thread; and

forwarding the virtual thread state corresponding to the first active thread to state update logic.

2. The method of claim 1, wherein the active thread state is received from a next thread (NT) multiplexer.

3. The method of claim 1, wherein the active thread state is received by a virtual state reload multiplexer of the virtual state mechanism.

4. The method of claim 1, wherein the generating of the virtual thread state is performed by a state register of the virtual state mechanism.

5. The method of claim 1, further comprises for every cycle, reloading the virtual thread state corresponding to the first active thread using the virtual state reload multiplexer for as long as the first active thread remains active.

6. The method of claim 1, further comprising:

maintaining the virtual thread state using the virtual thread mechanism until the first active thread becomes inactive and a second thread becomes active; and

detecting the activation of the second active thread.

7. The method of claim 6, further comprising:

receiving active thread state of the second active thread if the activation of the second active thread is detected;

updating the virtual thread state in accordance with the active thread state of the second active thread; and

forwarding the virtual thread state corresponding to the second active thread to the state update logic.

8. The method of claim 1, further comprising:

detecting an uncommon event in a path between the state update logic and the first active thread; and

performing state restoration using a state restoration multiplexer corresponding to the first active thread.

9. A method, comprising:

receiving active thread state of a first active thread using a virtual state mechanism;

generating virtual thread state in accordance with the active thread state of the first active thread; and

maintaining the virtual thread state corresponding to the first active thread until the first active thread becomes inactive.

10. The method of claim 9, wherein the maintaining of the virtual thread state comprises reloading the virtual thread state corresponding to the first active thread using a virtual state reload multiplexer of the virtual state mechanism until the

first active thread becomes inactive and a second thread becomes active.

11. The method of claim 9, wherein the receiving of the active thread state is performed by the virtual state reload multiplexer.

12. The method of claim 9, wherein the generating of the virtual thread state is performed by a state register of the virtual state mechanism.

13. The method of claim 9, further comprises forwarding the virtual thread state corresponding to the first active thread to state update logic.

14. The method of claim 10, further comprising:

detecting the activation of the second active thread;

receiving active thread state of the second active thread if the activation of the second active thread is detected;

updating the virtual thread state in accordance with the active thread state of the second active thread; and

forwarding the virtual thread state corresponding to the second active thread to the state update logic.

15. A processor, comprising:

a virtual state reload multiplexer to receive active thread state of a first active thread; and

a state register to generate virtual thread state in accordance with the active thread state of the first active thread.

16. The processor of claim 15, wherein the state register is further to forward the virtual thread state corresponding to the first active thread to state update logic.

17. The processor of claim 15, wherein the virtual state reload multiplexer is further to:

maintaining the virtual thread state comprising reloading the virtual thread state corresponding to the first active thread until the first active thread becomes inactive and a second active thread becomes active; and'

receive active thread state of the second active thread if the activation of the second active thread is detected

18. The processor of claim 15, the state register is further to:

detect the activation of the second active thread;

update the virtual thread state corresponding to the second active thread; and

forward the virtual thread state corresponding to the second active thread to the state update logic.

19. A system, comprising:

a storage medium; and

a processor coupled with the storage medium, the processor having

a virtual state reload multiplexer to receive active thread state of a first active thread; and

a state register to generate virtual thread state in accordance with the active thread state of the first active thread.

20. The system of claim 19, wherein the state register is further to forward the virtual thread state corresponding to the first active thread to state update logic.

21. The system of claim 19, wherein the virtual state reload multiplexer is further to:

reload the virtual thread state corresponding to the first active thread until the first

active thread becomes inactive and a second active thread becomes active; and

receive active thread state of the second active thread if the activation of the second active thread is detected

22. The system of claim 19, the state register is further to:

detect the activation of the second active thread;

update the virtual thread state corresponding to the second active thread; and

forward the virtual thread state corresponding to the second active thread to the state update logic.

23. A machine-readable medium having stored thereon data representing sets of instructions, the sets of instructions which, when executed by a machine, cause the machine to:

receive active thread state of a first active thread using a virtual state mechanism;

generate virtual thread state in accordance with the active thread state of the first active thread; and

forward the virtual thread state corresponding to the first active thread to state update logic.

24. The machine-readable medium of claim 23, wherein the sets of instructions which, when executed by the machine, further cause the machine for every cycle, reload the virtual thread state corresponding to the first active thread using the virtual state reload multiplexer for as long as the first active thread remains active.

25. The machine-readable medium of claim 23, wherein the sets of instructions which, when executed by the machine, further cause the machine to:

maintain the virtual thread state using the virtual thread mechanism until the first active thread becomes inactive and a second thread becomes active; and

detect the activation of the second active thread.

26. The machine-readable medium of claim 23, wherein the sets of instructions which, when executed by the machine, further cause the machine to:

receive active thread state of the second active thread if the activation of the second active thread is detected;

update the virtual thread state in accordance with the active thread state of the second active thread; and

forward the virtual thread state corresponding to the second active thread to the state update logic.

27. The machine-readable medium of claim 23, wherein the sets of instructions which, when executed by the machine, further cause the machine to:

detect an uncommon event in a path between the state update logic and the first active thread; and

perform state restoration using a state restoration multiplexer corresponding to the first active thread.

28. A machine-readable medium having stored thereon data representing sets of instructions, the sets of instructions which, when executed by a machine, cause the machine to:

receive active thread state of a first active thread using a virtual state mechanism;

generate virtual thread state in accordance with the active thread state of the first active thread; and

maintain the virtual thread state corresponding to the first active thread until the first active thread becomes inactive.

29. The machine-readable medium of claim 28, wherein the sets of instructions which, when executed by the machine, further cause the machine to reload the virtual thread state corresponding to the first active thread using a virtual state reload multiplexer of the virtual state mechanism until the first active thread becomes inactive and a second thread becomes active.

30. The machine-readable medium of claim 29, wherein the sets of instructions which, when executed by the machine, further cause the machine to:

detect the activation of the second active thread;

receive active thread state of the second active thread if the activation of the second active thread is detected;

update the virtual thread state in accordance with the active thread state of the second active thread; and

forward the virtual thread state corresponding to the second active thread to the state update logic.